FOREIGN OBJECT DEFINITION INFORMATION REPOSITORY

1. FIELD OF THE INVENTION

The present invention relates generally to object-oriented or object-based distributed object management technology, and more specifically to a repository for foreign object definition information, a gateway for manipulating foreign objects using such foreign object definition information, and a network management system using such a gateway.

2. BACKGROUND

Object-oriented and object-based programming techniques encapsulate computer programs and data in objects which separate implementation details from the contractual interface through which the objects are used. An object's interface defines the object's type, and is the view of the object exposed and accessible from outside the object. The interface is a usually a listing of the operations and attributes that an operation provides. For example, a typical object interface includes the object's methods and their signatures together with any externally accessible fields. The details of the object's implementation other than those reflected in the interface are hidden. See Booch, Object-Oriented Analysis and Design, Addison-Wesley (1994). Objects are invoked through their interface.

Because an object invocation must conform to the object's interface, any entity seeking to invoke an object must have access to interface definition information for the object at the time it makes the invocation.

Interface information may be acquired either statically or dynamically. Statically acquired interface information is typically compiled or built into an invoking entity (usually another object), and cannot be changed without recompiling or modifying that entity. Dynamically acquired interface information, on the other hand, is typically acquired at run-time, and permits the construction of an invocation of an object having an interface that was unknown to the invoking entity prior to run-time.

Interface definition information may be acquired dynamically by interrogating objects for interface information about themselves, as for example through the Java introspection mechanism, or by acquiring the interface information from some other object or program as in the case of the CORBA Interface Repository.

Object interfaces are specified in a variety of ways. In an object-oriented language, an interface may be defined concretely by simply implementing it in an object. Alternatively, the interface may be specified apart from any object or class implementing the interface. Some object-oriented languages, such as Java, include an interface declaration keyword for abstractly defining interfaces. Abstract or virtual classes can also be used for this purpose in some languages.

A higher degree of abstraction and implementation independence may be achieved by defining interfaces in a language-independent interface definition notation. An interface defined this way may typically be implemented in more than one language, and objects so implemented in one language may typically invoke services of objects implemented in another language. In a distributed object-oriented environment, the ability to connect objects implemented in a variety of languages is a significant benefit.

One distributed object-oriented environment with multiple-language support is the Common Object Request Broker Architecture ("CORBA") specification published by the Object Management Group. CORBA interfaces are defined abstractly using the CORBA Interface Definition Language ("CORBA IDL"), a declarative notation for defining interfaces. CORBA IDL is the means by which clients learn what operations are available from an object, and how they should be invoked.

The CORBA specification includes language mappings by which an interface defined in CORBA IDL is translated into "stub" code implementing the interface for an invoking object, and "skeleton" code implementing the interface on the invoked object. CORBA IDL language mappings exist for a variety of languages including Smalltalk, C++ and Java. See The Common Object Request Broker: Architecture and Specification, Revision 2.0, July 1995, updated July 1996.

Another distributed object-oriented environment supporting language-independent object definitions is included in the Open System Interconnection (OSI) network management standards published by the International Organization for Standardization (ISO). The Common Management Information Protocol (CMIP) and the Common Management Information Service Element (CMISE) defined in these standards incorporate a distributed object model for network management. This model is described in part in CCITT Recommendation X.722 (1992), "Information Technology -- Open Systems Interconnection - Management Information Services - Structure of Management Information: Guidelines for the Definition of Managed Objects." Objects defined in accordance with Recommendation X.722 are referred to in this specification as "GDMO objects."

In the OSI world, objects are specified in GDMO/ASN.1. "ASN.1" refers to Abstract Syntax Notation One, a complex type definition notation. GDMO/ASN.1 object definitions are based on a set of templates used to describe managed objects used for network management and control. Among other object information, ASN.1 is used to define transfer syntaxes which can be used to invoke GDMO objects. ASN.1 is specified in ITU-T Recommendation X.208, "Specification of Abstract Syntax Notation One (ASN.1)".

When objects defined in one scheme (such as CORBA, for example) are used to manipulate (for example to invoke or instantiate) objects defined in another scheme (such as GDMO/ASN.1, for example), object definition information, including interface information must be somehow acquired across definition schemes.

Object definition information may be acquired across definition schemes by translating definitions from one scheme into the other. However, a syntactic or semantic mismatch between the schemes may make the translation or its results cumbersome or unworkable.

GDMO/ASN.1, for example, supports a richer variety of types than does CORBA IDL. Translation into CORBA IDL from ASN.1 often leads to unmanageable CORBA IDL definitions, and very large executables. Also, CORBA IDL translated from GDMO/ASN.1 is typically incomplete because information is lost during the translation process. The resulting collection of managed object definition information may not contain all the required information to perform CMISE operations.

There is therefore a need to provide a system for permitting objects having definitions in one definition notation to access object definitions specified in a different object definition notation in a manner that does not cause information to be lost in the process.

Accordingly, it is one objective of the present invention is to provide a first set of object definition information specified in a first notation to objects specified at least in part in a second notation without translating the first object definition information into the second object definition notation.

Another objective of the invention is to provide a means by which objects specified at least in part in a first object definition notation can invoke objects specified at least in part in a second object definition notation.

A further objective of the invention is to provide a means by which objects can instantiate objects defined at least in part in a foreign object definition notation.

Another objective of this invention is to provide a metadata repository by which a objects can use interfaces defined in a first object definition notation to discover object definitions specified in a second object definition notation.

3. **SUMMARY**

The invention accomplishes these and other objectives by encapsulating foreign object definition information in encapsulator objects having native interfaces. The encapsulated foreign object definition information need not be translated into native definition notation, but may be discovered by interrogating the encapsulator objects using their native interfaces.

Encapsulator objects can include all information contained in GDMO/ASN.1 specifications, in any suitable format. Encapsulated information may be discovered by invoking the encapsulator objects on their CORBA interfaces without translating the encapsulated specifications into CORBA IDL.

The invention also encompasses the use of encapsulator objects in a foreign object definition information repository. Such a repository includes supporting objects that provide additional services for run-time discovery of foreign object definition information, such as searching for specific types of foreign object definitions.

The encapsulator objects and metadata repository of the invention may be used to construct a dynamic gateway that does not require compile-time information of object definitions of objects using its services. All information necessary for adaptation of the dynamic gateway can be provided through the metadata repository. Such a gateway may be provided as an off-the-shelf application for deployment in a variety of environments.

4. **BRIEF DESCRIPTION OF THE DRAWINGS**

Fig. 1 is a block diagram of an object definition metadata repository showing encapsulator and support object types.

Fig. 2 is a block diagram of a dynamic CORBA/CMIS gateway comprising a object definition metadata repository.

Fig. 3 is a block diagram comprising a greatly simplified representation of a collection of encapsulator objects and references corresponding to a completely specified GDMO/ASN.1 object definition.

5. DETAILED DESCRIPTION OF THE INVENTION

In one embodiment, the invention comprises a method and system for encapsulating object definition information expressed in a first notation in encapsulator objects defined at least in part in a second, typically different notation. In a preferred embodiment, the encapsulator objects expose interfaces defined in the second notation. This specification refers to such a first notation as foreign notation, and such a second notation as native notation. Corresponding objects, object definitions, interfaces, interface information and interface definitions are also referred to as "foreign" and "native" respectively.

In a preferred embodiment, encapsulator objects are created by a parser that parses foreign notation. For example, in one preferred embodiment, the parser accepts foreign object definition information expressed in foreign notation and instantiates one or more encapsulator objects for each rule (such as a production) in a grammar that corresponds to the syntactic structure defined in the foreign notation. The resulting collection of encapsulator objects reflects the syntactic structure of the foreign object definition information. The encapsulator objects expose native interfaces that may be interrogated to discover the foreign object definition information they encapsulate.

Standard compiler generation tools such as lex and yacc may used to create a parser, such as a parser for GDMO/ASN.1 specifications, in accordance with the present invention. The system may in addition resolve ISO object identifiers into their integer components, perform semantic checking and resolve informal references to other specifications.

For example, an informal reference such as "CCITT Recommendation X.721:1992" may be resolved to a file name such as "x721.gdmo". Textual references are resolved to CORBA object references. For example, "CCITT Recommendation X.721:1992:top" would be resolved to a pointer to the CORBA object encapsulating the first specification in Recommendation X.721. An ISO object identifier such as {joint-iso-ccitt ms(9) smi(3) part2(2) asn1Module(2) 1} would be resolved to {2 9 3 2 2 1}.

The parser may be included in an object factory (an object which instantiates and initializes other objects) which instantiates CORBA objects corresponding to the nonterminals of the syntax defined by GDMO/ASN.1 specifications. The resulting CORBA objects may be interrogated by other CORBA objects through their CORBA interfaces to discover the encapsulated GDMO object definitions. No translation of the encapsulated GDMO object information into CORBA IDL is required, avoiding the problems posed by the differences between CORBA and GDMO syntax and semantics.

In a preferred embodiment, the encapsulator objects have predefined native interfaces. This permits invocation of the encapsulator objects without run-time discovery of their native interfaces. An alternative embodiment includes encapsulator objects that have native interfaces that are not predefined. Such encapsulator interfaces may depend upon the foreign interface definitions which they encapsulate, or be determined by other information available at the time the encapsulator objects are instantiated. Still another embodiment includes the use of pseudo-objects accessible through, for example, the CORBA Dynamic Skeleton Interface (DSI). Such a pseudo-object embodiment could, for example, parse foreign object definitions at run-time in response to invocations on a pseudo-object, or store foreign object definition information in a database.

The type structure of the encapsulator objects may reflect the type structure of the foreign interfaces which they encapsulate. For example, in a preferred embodiment, a CORBA GDMO repository includes CORBA encapsulator object types corresponding to GDMO templates, including *Managed Object Class Template*(101), *Package Template* (102), *Attribute Template* (103), *Attribute Group Template* (104), *Notification Template* (105), *Action Template* (106), *Parameter Template* (107), *Name Binding Template* (108) and *Behavior Template* (109). In addition, a CORBA encapsulator object for a *Module* ASN.1 container type is included (110), as shown in Fig. 1.

In a preferred embodiment, the collection of encapsulator objects has the form of a metadata repository. In such an embodiment, support objects provide services for the discovery of encapsulator objects. For example, a foreign object definition metadata repository could provide a class having a native interface that accepts invocations for all encapsulator types, and which parses the invocations using the encapsulator objects. Other services, such as for iteration through the encapsulator objects in the repository, or sophisticated query and/or scoping services may be provided, permitting the discovery of encapsulator objects having only a certain type or property.

A foreign object definition metadata repository may be used in applications requiring dynamic discovery of foreign object definitions. An example of such an application is a CORBA based GDMO browser, which would use predefined CORBA interfaces to discover and present information from the repository, and could permit manipulation on the fly of GDMO objects having definitions encapsulated in encapsulator objects visible to the browser. Other examples such as a dynamic gateway or code generator are described in greater detail below.

In one preferred embodiment, the encapsulator repository has the form of a GDMO repository, and a containment structure as shown in Fig. 1. In the GDMO repository of this embodiment, a CORBA GDMO Repository object (111) contains one or more CORBA GDMO Document objects (112). Each Document object in turn contains one or more CORBA objects of types corresponding to GDMO template and ASN.1 Module types.

By using the services of the Repository and Document supporting objects, CORBA client objects can discover CORBA encapsulator objects encapsulating particular GDMO template types and/or originating in a particular GDMO definition document. For example, the Repository object (111) could be interrogated to discover all available objects encapsulating GDMO Attribute Group template types. Or a Document object corresponding for example to CCITT Recommendation X.721 could be interrogated to discover all objects encapsulating GDMO Action template types specified in that Recommendation.

Encapsulator objects, parsers of object definition information and object factories for instantiating encapsulator objects, as well as supporting objects for discovery of encapsulator objects and applications using these structures may be constructed in a variety of ways that will be readily apparent to those of ordinary skill working in the distributed object and network management fields.

Encapsulator objects may be used to advantage in a number of applications. In particular, gateways for manipulating foreign objects may be readily implemented through the use of encapsulator objects to construct invocations on foreign objects.

In a preferred embodiment, a dynamic gateway uses a foreign object definition metadata repository to manipulate foreign objects at run time in response to requests on native interfaces for services requiring use of those foreign objects.

In one preferred embodiment, a dynamic CORBA/CMIS gateway is implemented through the use of a foreign object definition repository as shown in Fig. 2. The gateway includes both a CORBA server (201) and a CMIS manager (202) interoperating as a single application (203). The CORBA server (201) exports a CORBA interface (204) through which one or more CORBA clients such as (205) may utilize CMIS services and CMIP (208) to invoke services of a CMIS agent (206). Encapsulator objects in a GDMO/ASN.1 repository (207) are referenced by the CORBA server (201) to provide the object definition information required by the CMIS manager to manipulate one or more CMIS agents such as (206).

The dynamic CORBA/CMIS gateway (203) is a generic application that can accept a variety of requests from CORBA objects such as (205) for CMIS operations without compile-time knowledge of the GDMO/ASN.1 definitions for the CMIS operations requested. By using CORBA DSI, the generic gateway can accept invocations on CORBA IDL interfaces that depend on the CMIS operations desired and parse those invocations using the GDMO/ASN.1 repository. The generic gateway can thus be deployed off-the-shelf in a variety of environments by encapsulating the CMIS operation dependent information for each environment in the foreign interface repository.

In this embodiment, upon receipt of a request using DSI for a CMIS operation from a CORBA object, the dynamic gateway invokes a root repository object using managed object class name information included in the request. The root repository object returns a reference to an encapsulator object corresponding to a managed object class template specified by the name. The managed object class template object corresponds to the root of a subtree of encapsulator objects that fully defines the class. The subtree objects correspond to GDMO template types.

The managed object class object is invoked using additional information (for example, attribute information) from the request, and returns one or more references to encapsulator objects in the subtree and/or one or more terminals representing symbols in the parsed request. Any referenced objects are in turn invoked, and the subtree is traversed in this manner until the request is resolved into a collection of CORBA object references that corresponds to a completely specified GDMO/ASN.1 object definition. Operation name information in the request is parsed in a similar manner. The specified definitions are then used to construct the requested CMIS operation invocation. Fig. 3 represents a simplified representation of such a collection of encapsulator object references. A typical collection of references in practice would be far more complex.

The dynamic gateway of this embodiment uses the CORBA Dynamic Skeleton Interface to accept CORBA invocations including CORBA IDL definitions not available to the gateway at compile time. The gateway uses the predefined CORBA interfaces of the GDMO metadata repository to discover the actual parameters being passed by the invocation as well as how to use those parameters to construct the requested CMIS operation invocation. The gateway can thus accept invocations having CORBA interfaces that depend on the CMIS operation requested without compile-time knowledge of either the complete CORBA requesting interface or the CMIS operation. The gateway can therefore be an off-the-shelf application adaptable through the repository to wide variety of circumstances.

For example, in a typical network management situation, a variety of devices such as routers, switches, adapters, modems, printers and other computer and communication equipment are connected to the managed network. Instances of managed object classes are used to control these devices, and will have object definitions that vary depending on the particular devices connected to the managed network. By placing the managed object definition information into encapsulator objects in accordance with the present invention, a dynamic gateway may be deployed over a variety of managed networks without modification to the gateway software.

Many other applications for foreign object definition encapsulator objects exist. One such application is a lightweight code generator. Such a code generator may be used for the creation of classes that may directly invoke foreign objects. For example, a CORBA code generator could interrogate a GDMO repository to construct Java or C++ code for directly invoking GDMO objects. It will be apparent to one of ordinary skill that invention is not limited to the specific embodiments set forth here for purposes of illustration but is applicable broadly to a wide variety of applications involving object manipulation across definition schemes, all as set forth in the claims.